Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2005 Proceedings

Americas Conference on Information Systems (AMCIS)

2005

Towards an Understanding of Model Driven Process Configuration and its Support at Large

Alexander Dreiling
University of Munster, alexander.dreiling@ercis.de

Mervin Chiang
Queensland University of Technology, mervin@mail.com

Michael Rosemann
Queensland University of Technology, m.rosemann@qut.edu.au

Wil M.P. van der Aalst
Eindhoven University of Technology, w.m.p.v.d.aalst@tm.tue.nl

Towards an Understanding of Model-Driven Process Configuration and its Support at Large*

Alexander Dreiling
University of Münster, Germany
alexander.dreiling@ercis.de

Mervin Chiang
Queensland University of Technology, Australia
mervin@mail.com

Michael Rosemann
Queensland University of Technology, Australia
m.rosemann@qut.edu.au

Wil M. P. van der Aalst
Eindhoven University of Technology,
The Netherlands
w.m.p.v.d.aalst@tm.tue.nl

ABSTRACT

The common presupposition of Enterprise Systems (ES) is that they lead to significant efficiency gains. However, this is only the case for well-implemented ES that meet organisational requirements. The list of major ES implementation failures is as long as the list of success stories. We argue here that this arises from a more fundamental problem, the functionalist approach to ES development and provision. As long as vendors will continue to develop generic, difficult-to-adapt ES packages, this problem will prevail because organisations have a non-generic character. A solution to this problem can only consist in rethinking the way ES packages are provided. We propose a strict abstraction layer of ES functionalities and their representation as conceptual models. ES vendors must provide sufficient means for configuring these conceptual models. We discuss in this paper what generic situations can occur during process model configuration in order to understand process model configuration in depth.

Keywords (Required)

Enterprise Systems, Organisations, Process Configuration, Requirements

INTRODUCTION

Enterprise Systems (ES) are in an area of conflict between business and IT. The sole purpose of an ES is to contribute to the effectiveness and efficiency of activities in organisations. For instance, efficient access of data or control—or automation—of processes are two exemplary basic functionalities on which concepts such as customer relationship management or supply chain management are built. The area of conflict arises from the fact that organisations, on the one hand, are highly specific and ES, on the other hand, are highly generic. Configuration is then seen as a structured process that is applied to a generic package which transforms the generic package into an organisation-specific one. In other words, a structure (generic package) is transformed within a structured process (configuration) to aid something supposedly less-structured (organisation) which in itself poses problems.

The concept of ES is subject to academic discussion for several years now. Examples of recent contributions (typically under the name of Enterprise Resource Planning – ERP) cover amongst others the definition of ERP (Klaus et al., 2000), critical success factors of ERP Systems (Akkermans and van Helden, 2002, Holland and Light, 1999), modelling within the context of ERP (Dalal et al., 2004), and possible developments of ERP in the future (Markus et al., 2000).

Organisation theory, on the other hand, provides a scattered picture as to what an organisation is. More specifically, modern explanations of organisations (e.g., Cyert and March, 1963, March and Simon, 1958) coexist with post-modern approaches (e.g., Boje et al., 1996, Hatch, 1997). This manifests, for instance, in management structures. Whereas the formal organisations is the *Gestalt* of an organisation in modern approaches and diverging power structures are treated as an abnormality, this is the opposite in post-modern approaches. The formal organisation and with it the management structure is seen as untenable and the enabler for ineffectiveness and inefficiencies. Whereas the functions of the management were clear and precise in classic and modern times (Barnard, 1938, Koontz and Cyril, 1968), postmodernist views, e.g., reject the notion of *decision* as what was perceived as one of the basic functions of an executive (Chia, 1996, Gephart Jr., 1996).

However, common presuppositions in IT and IS research typically do not transcend the ideas outlined by modern organisation theory. One of the consequences certainly is that generic packages without sophisticated means of their adaptation to the requirements of an organisation are meaningless. If organisation theory failed to deliver a concise explanation of organisations in a functionalist manner (Burrell and Morgan, 1979), how can we then assume that a generic package will serve the needs of an organisation? We argue here that this is not possible because functionalist approaches oversimplify. Rather, ES provision needs to be rethought. The ever increasing need for flexibility and adaptability of software has been discussed for a significant time now in IT and IS. Typically this is associated with the fact the business requirements change rapidly. The first change, however, is the one of a "vanilla" installation to the organisation-specific one. Especially the field of Workflow Management which implies a separation of process logic from application logic led to progress and theoretically and practically makes software more flexible.

It is within this discussion that the importance of a strict abstraction layer separating application logic from its representation—a set of conceptual models—exists. With regard to the discussion in this section, changes to the behaviour of the ES package must only be possible by changing conceptual models. This alters the common presupposition of reference modelling significantly. Whereas reference modelling typically is perceived as a prescriptive way of supporting organisations, it is in the sense of this research a description of an ES package in its standard form. Hence, a reference modeller cannot "objectively" prescribe how an organisation should conduct operations, he can only model any way, perhaps a statistically more frequently occurring one, but still any way. The ES package must then allow for adapting the models in an efficient an intuitive way in order to change the behaviour of the software package accordingly.

This proposed abstraction layer of models built on top of the application logic and its ties to the software will lead to a different set of research questions than those commonly addressed in IS and IT research. For instance, it is not necessarily meaningful to measure the gap between an organisation and an ES package (if this is possible at all) but rather to evaluate how intuitively the ES package can be adapted within the organisation. Also, the means of adaptation or configuration using models need to be examined thoroughly.

In this paper we describe how we address model configuration within a collaborative research project with a major ES vendor (reference to be added). We pick the process perspective and describe generic configuration patterns within process models in the next section. We will distinguish between semantic and syntactic patterns and explain where in a configuration project the single types of patterns apply. This section is followed by a brief description of implemented prototypes that aid in assisting a configuration project. We will then conclude with a short outlook and summary.

MODEL-DRIVEN PROCESS CONFIGURATION OF ENTERPRISE SYSTEMS

Configuration has been discussed to a certain extent in academia, covering the adaptation of software packages in general (Gibson et al., 1984, Lucas Jr. et al., 1988), taxonomies for configuration (Brehm et al., 2001), or methods for model configuration (Soffer et al., 2003). Configuration and customisation are often used interchangeably Merriam-Webster's Collegiate Dictionary defines configuration as the "relative arrangement of parts or elements" whereas customising is defined as "to build, fit, or alter according to individual specifications" (Merriam-Webster, 2003). With these definitions in mind we can only perform reconfiguration (alteration of relative arrangement of parts or elements within enterprise systems) or customisation (alteration of enterprise systems in order to meet the specification of the enterprise). The latter includes alterations of program code which, we do not pursue in our research. We are rather concerned with the configuration of enterprise systems. For the purpose of this paper, we define (re-)configuration of an enterprise system as the process of aligning business aspects such as functions, information, processes, or organisation with generic enterprise systems in order to meet the business requirements of the enterprise in the most efficient way. For the sake of simplicity we will use the term configuration instead of reconfiguration from here on.

In order to configure an ES for an organisation using a conceptual model-driven approach, we have to understand how a configured set of models can be derived from a reference model. Since we focus on process configuration for the purpose of this paper, we have to understand what generic situations can occur during configuration in order to address these explicitly or implicitly during reference modelling. In this section, we will briefly discuss the three dimensions *semantic configuration* which refers to configuration decisions from a business perspective, *syntactic configuration* which refers to rules for maintaining the syntactic correctness during configuration, and *pragmatic configuration* which deals with enhancing the model quality in order to cater for issues related to understandability during a configuration project.

Semantic Configuration Patterns

There is a pressing need to configure conceptual models as a mirror of the intended behaviour of the ES for implementation within an organisation. The results from this phase in the project would be a set of configured conceptual models. This phase

of the project is called *configuration time* (Rosemann and Aalst, 2003). It is the moment in time where configuration decisions need to be made and build time models are derived from (not necessarily executable) reference models.

Consequently, *semantic configuration patterns* allow for a formalised understanding of how configuration occurs from a business perspective and help to identify possible configuration alternatives. Semantic configuration patterns are defined as patterns which depict a configuration scenario and highlight the potential implementation alternatives that are available. The focus was on functions, connectors, and control flow elements of the model. With the help of *workflow patterns* (Aalst, 2003), by analysing all of their components that can be configured and examining all possible build time scenarios, nine semantic configurable patterns have been developed. These nine patterns form the semantic foundation that governs our model-driven process configuration. They can be classified into function-related and control flow-related patterns. All configuration patterns are depicted in Event Driven Process Chain (EPC) notation (Scheer, 2000) which is a commonly accepted modelling technique for modelling processes from a business perspective. The three function-related patterns are (Table 1 lists all function-related pattern with their related configuration choices):

- *Optionality* (choices *on, off, optional*): during configuration time, a configurable function of an EPC can be switched *on*, *off* or *optional* (choice deferred to run time and included in build time model).
- *Interleaved Parallel Routing*: order of execution for a number of functions within a process can be configured, all functions must be executed, none of them at the same time.
- *Sequence Inter-relationships*: two or more functions may be dependable on each other during configuration.

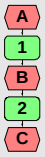
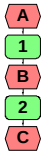
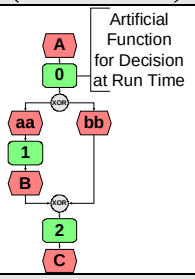
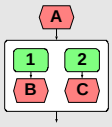
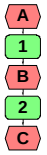
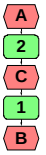
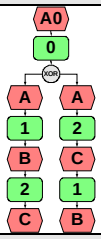
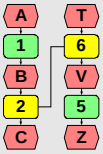
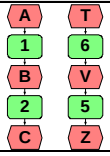
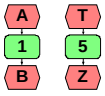
Semantic Configuration Pattern	Depiction	Build Time Configuration Possibilities		
Optionality		Combination *(Switched ON)*	Partial *(Switched OFF)*	Conditionally Skipped *(Switched OPT)*
			Syntactically, events A and B can trigger the process. The semantic assumption here is that event A triggers it. 	
Interleaved Parallel Routing		Assuming semantic definition of Event A is to initiate the process and either Event B or C terminates the process.		
		Sequence 1 fixed at build time 	*Sequence 1 fixed at build time* 	*Sequence 1 fixed at build time* 
Inter-relationships		Assuming that semantically Events C and V are output events of Function 2 and 6.		
		Sequences 1 		*Sequences 2* 

Table 1. Configuration Pattern of Optionality, Interleaved Parallel Routing, and Inter-relationships

The second type of semantic configuration comprises six control flow-related patterns (Table 2 shows the splitting connectors only, their corresponding joins are similar). These patterns are:

- *Parallel Split* (choices *AND*): if an AND-split is configurable it cannot be changed itself (to, e.g., an OR or XOR connector). The number of outgoing branches however, can be configured. The AND connector itself thus stays and AND connector, yet its semantics can be changed by restricting the branches which must be executed in parallel.
- *Synchronisation*: handles the merging of branches from the configurable AND connector. As with the corresponding split, the connector itself cannot be changed but incoming arcs can be removed.
- *Exclusive Choice*: explains the instances where an XOR is configurable during configuration time. The split can either remain as an XOR split which defers the decision to run time (i.e., it is kept in the build time model) or the exclusive choice can be made at configuration time which transfers the model into either one of two sequences (in case of two outgoing branches).
- *Simple Merge*: corresponds to Exclusive Choice.
- *Multi Choice*: is the most powerful configurable split in terms of expressiveness. It can be configured into either one split or the sequences that result from either one of the outgoing arcs (in case of two outgoing arcs).
- *Synchronising Merge*: corresponds to Multi Choice.

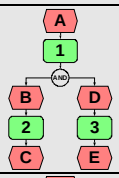
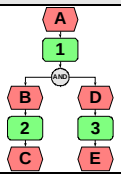
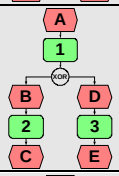
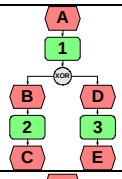
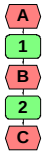
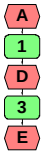
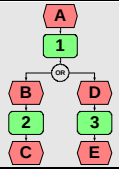
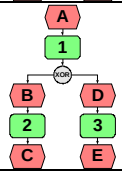
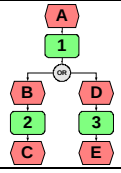
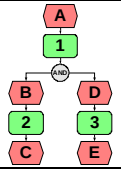
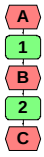
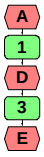
Semantic Configuration Pattern	Depiction	Build Time Configuration Possibilities				
		XOR	OR	AND	Sequence	
					Sequence 1	*Sequence 2*
Parallel Split		*n.a.*	*n.a.*		*n.a.*	*n.a.*
Exclusive Choice			*n.a.*	*n.a.*		
Multi Choice						

Table 2. Semantic Configuration Patterns of Parallel Split, Exclusive Choice and Multi Choice

Syntactic Configuration Patterns

Having now seen the semantic dimension of the model-driven process configuration approach, one must still consider maintaining syntactically correct control flow within the EPC. For example, if a function was turned off, based on the meta model of EPCs, the process model will be syntactical inconsistent: assuming the control flow is reconnected where the function is missing, two events follow each other. Inadvertently, the step beyond semantic configuration is the need for re-establishing syntactical correctness and consistency. For this reason seven syntactic patterns were developed.

These patterns were outlined to cater for each conceivable scenario of the control flow in all levels of granularity. Each pattern is identified with the sequence it deals with. We must limit the discussion of syntactical configuration patterns to three representative examples within this paper. The first pattern, *Event-Function-Event (E-F-E)*, enables three conceivable scenarios, i.e., to remove the preceding event with a function, to remove its succeeding event, or to remove both events and replace them with a new one. The pattern *Join Connector-Function-Event (JC-F-E)* handles a function that has a preceding join connector and an event receding it. It is important to note that in this case either the succeeding event will be removed along with a function or the preceding ones before the join. Additionally, both the succeeding event and the set preceding events can be updated if semantically necessary. The syntactic configuration pattern called *Join Connector-Function-Split Connector (JC-F-SC)* targets the configurable function that is framed by a join connector preceding and a split connector as receding it. The two sets of succeeding or preceding events can be either removed with the function or altered if semantically

necessary. Table 3 is an explained representation of the patterns and its conceived scenarios. To exemplify a clearer depiction, the configurable function in the pattern has a thick line surrounding it.

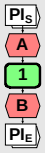
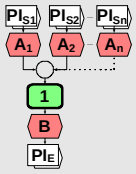
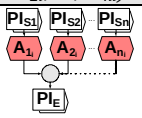
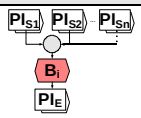
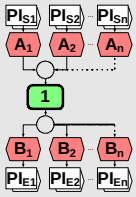
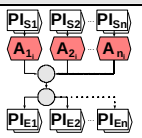
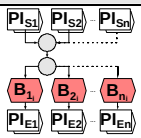
Syntactic Configuration Pattern	Depiction	Build Time Configuration Possibilities		
E-F-E		Scenario 1 *(B is removed with 1)*	Scenario 2 *(A is removed with 1)*	Scenario 3 *(Both A and B are removed with 1 and replaced with X)*
				
JC-F-E		Scenario Group 1 *(B_1 is removed with 1, A_1, A_2,…, A_n are updated to A_{1i}, A_{2i},…, A_{ni})*	Scenario Group 2 *(A_1, A_2,…, A_n are removed with 1, B is updated to B_1)*	
				
JC-F-SC		Scenario Group 1 *(B_1, B_2,…, B_n are removed with 1, A_1, A_2,…, A_n are updated to A_{1i}, A_{2i},…, A_{ni}, this case is only possible with the AND connector)*	Scenario Group 2 *(A_1, A_2,…, A_n are removed with 1, B_1, B_2,…, B_n are updated to B_{1i}, B_{2i},…, B_{ni})*	
				

Table 3. Syntactic Configuration Patterns

Considering the Pragmatic Dimension

The third and final dimension for using Model-Driven Process Configuration is the need to consider the pragmatics of a conceptual model. Within the fields of Information Systems and Computer Science numerous process modelling languages have been developed over the last decades. These techniques vary in the degree as to which they are understandable to certain user groups such as management, business users, or technical users, i.e., they are of different pragmatic quality (Lindland et al., 1994). This arises from the fact that some process modelling languages depict business processes from a conceptual perspective with a focus on understanding and improving the process. In these cases, an intuitive comprehensiveness for a large number of users with typically limited modelling experiences is more important than the use of a language with high expressive power. Other modelling techniques describe a business process with the purpose to derive executable workflow models. Such techniques demand a high rigor in terms of their meta model, but are often only used by a limited number of experienced modellers. In summation, it seems to be valid that the more detailed the control flow of a business process needs to be outlined, the more complex the process modelling language needs to be, featuring several constructs for handling splits, joins, varying numbers of instances, etc.

In order to carry out Model-Driven Process Configuration, we obviously need to consider semantic and syntactic correctness. Conversely, there is also a similar need to maintain a pragmatic modelling language that is easily understandable and effectively used by less experienced modellers. Becker et al. (2002) describe a *configurative reference modelling approach* and introduce the concept of *representation variation* in order to change the visualisation of the models. Additionally, IDS Scheer's ARIS Toolset provides so-called *templates* that can be applied to models in order to change their presentation. Furthermore, this tool provides several lay-out options to enhance the presentation of the model. We consider these options as sophisticated and do not aim to develop new means for the purpose of pragmatically enhancing the derived set of models.

However, we have to address pragmatic issues for the reference model base when we want to configure processes on a large scale because both the sizes and amounts of process models that are necessary in order to describe sophisticated ES can become dramatically large. Thus, we need to have a simple and intuitive modelling language that requires a user to have minimum experience to pick up and yet wield sufficient logical complexity to drive configurations into the Enterprise

System. A number of concepts have been created and implemented to synergise the pragmatic dimension (easier usability) into the semantic and syntactic aspect of the proposed approach. To manifest the realisation of this new modelling approach, in the next section, we have identified and introduced a number of visual aids (Configurable EPC's *Configuration Attributes* and *Highlighted Perspectives*) and guided extensions (Configurable EPC's *Configuration Notations* and *Context Menus*) to accomplish this endeavour.

REALISATION OF THE MODEL-DRIVEN PROCESS CONFIGURATION OF THE ENTERPRISE SYSTEM

In order to provide a viable and accepted approach, each outlined dimension requires a corresponding functional solution. The synergy of these solutions leads to a realisation of a new modelling language, which is an extension to the currently static EPCs. The ARIS platform was chosen to implement and demonstrate an initial proof-of-concept prototype. In order to facilitate the semantic aspect of configuration, Configurable Event Driven Process Chains (CEPCs) have been developed (Rosemann and Aalst, 2003) by extending the traditional EPC notation (Scheer, 2000). In total, four Configurable Nodes and two Configurable Attributes have been added to the EPC in order to make it a configurable modelling technique. We have extended this list to a total of eight based upon the semantic configuration patterns. Configurable nodes are *Configurable Function*, *Configurable XOR*, *Configurable OR*, and *Configurable AND* (all denoted by the original symbol surrounded by a thicker line in order to make the difference visible). Configurable Attributes include *Guideline* (a configuration decision is being guided), *Routing Container* (blocks of EPCs must be included in an arbitrary order), or *Default* (a configuration decision has a default value which is taken if the user does not touch this part of the reference model).

Figure 1 contains an example that shows a configurable invoice verification process. Our example company identified *Consignment/Pipeline Settlement* as not necessary since they do not run any consignment warehouses. The guideline for *ERS* recommends keeping ERS in case there are long-term contracts with suppliers and goods and conditions are specified. The organisation identified some supplier-goods combinations where this is the case and keep ERS. Additionally, the organisation identified *Invoicing Plan Settlement* as necessary and keeps it, which in itself already would have led to keeping ERS since a requirement for Invoicing Plan Settlement is that ERS must remain in the model if Invoicing Plan Settlement remains in the model. The three configurable connectors are accepted as they are delivered in the reference model.

With new CEPC notations integrated into ARIS we were able to begin initial work on programming a demonstration prototype using the example. *Context Menus* were used for configuring the CEPCs. When a user wishes to make a configuration decision during configuration time, a simple right-click on the configurable object would allow the user to select an available alternative (ON, OFF or OPTIONAL) from the *Context Menu* that appears. Similarly, a right-click on a configurable connector would allow the user to select the available alternative connector to put in place (OR, AND, or XOR) with the similar menu. With this, the modeller can complete a Model-Driven Process Configuration task on one CEPC in three steps *Visual Appraisal* (refers to the Configurable Attributes for information), *Perform Configuration* (configuration of conceptual models based on organisational requirements), and *Commit Configuration* (saves newly configured models).

Visual Appraisal is primarily the *"studying" phase* during configuration time. Here, the prevailing pragmatic dimension is manifested in the forms of intuitive understanding information and easy identification of procedures presented to the modeller. Along with the use of introduced configuration attributes, we also implemented a visual aid called *Highlighted Perspectives*. Because of the magnitude of models that need to be configured in an extensive ERP implementation, an introduction of the context menu selection, though greatly reduces the manual modelling during configuration time, may not be sufficient. As a further annexe to this, *Highlighted Perspectives* was implemented to cope with the mass of CEPC objects that may appear on the screen. The following three procedures were written to highlight perspectives of the process model that the modeller may want to quickly refer:

- Show all *Mandatory Functions:* This procedure highlights all non-configurable functions that are critical to the execution of the process. In the example shown in Figure 1, the functions *Process Invoice*, *Release Invoice Manually*, and *Release Invoice Automatically* will be highlighted.
- Show all *Mandatory Configurable Decisions:* This procedure highlights all mandatory configuration decisions that the modeller has to make for the successful execution of the process. Referring to the same example, *Invoicing Plan Settlement (IPS)* and *Evaluated Receipt Settlement (ERS)* were configurable functions that were dimmed as critical configuration decisions that the modeller is required to attend to. Correspondingly, a critical configuration decision was necessary for the configurable OR connector that merges the events *Purchase order created*, *Service is accepted*, and *Goods receipt posted*.
- Show all *Model Dependencies:* This procedure supports the configuration pattern *Inter-relationships* in highlighting relationships between configurable functions or connectors to visually illustrate its dependencies. Taking the above example again, this time *IPS* and *ERS* will be highlighted and a link is shown to illustrate *IPS's* dependence on *ERS*.

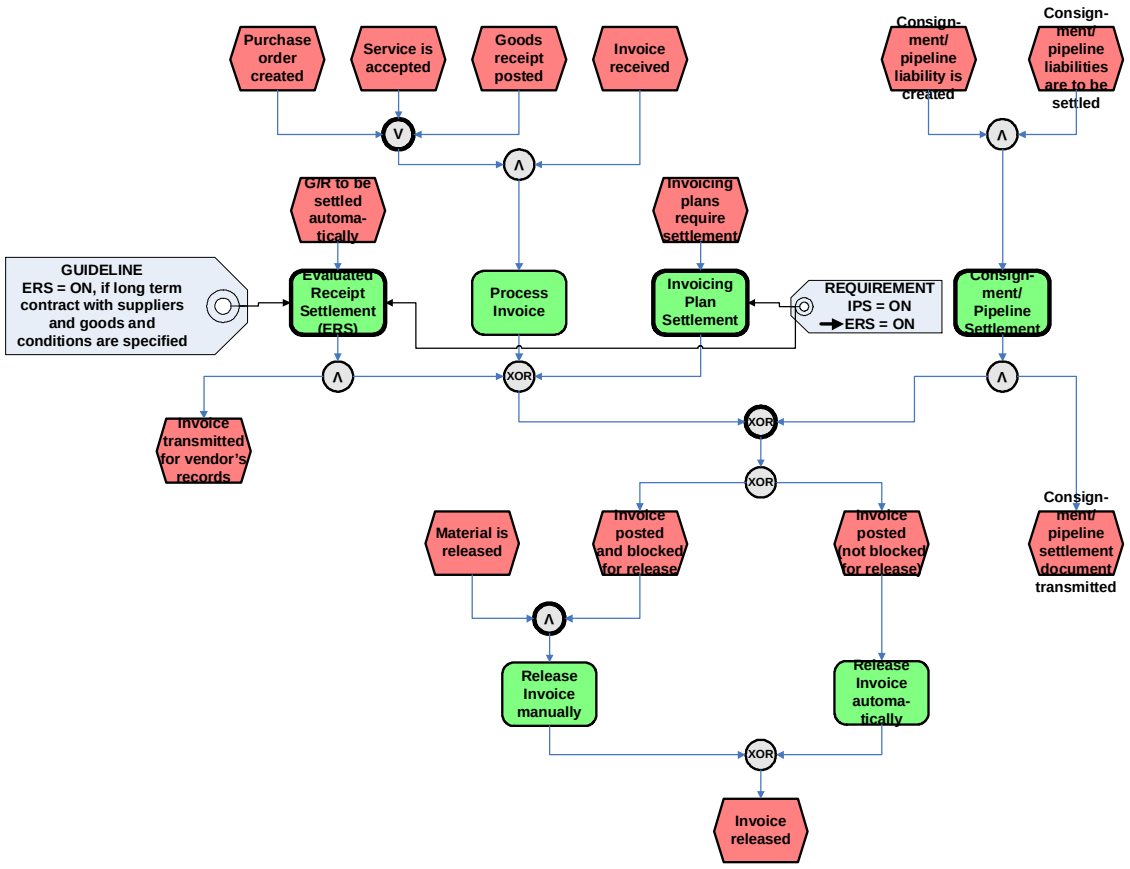


Figure 1. Example Configurable Process

To retain knowledge of what to highlight with which corresponding rule, a set of attributes were defined and attached to each object based on the CEPC's requirements. The proof-of-concept prototype then picks up these attributes and displays the modeller's corresponding choice by highlighting the respective objects and dimming the other objects in contrast.

Perform Configuration represents the main activity of a model-driven process configuration task. This phase encapsulates a collage of repetitive sub-steps to accomplish the task. These sub-steps can be classified into this successive and repeatable order:

1. *Perform Semantic Configuration:* This step involves an actual execution of a business decision that is manifested during configuration time. This means a selection from the context menus of configurable objects within the CEPC is performed here.
2. *Perform Semantic Follow-up Configuration:* Once a configuration decision is executed (Function-related: ON, OFF, or OPTION, or, Control Flow-related: OR, XOR, or AND), based on the semantic pattern scenarios outlined previously, the modeller has to ensure that the CEPC is semantically consistent. This results, e.g., in removing depending functions with functions that have been removed during the first step.
3. *Perform Syntactic Configuration:* Consequently, as the modeller is making the decisions and putting in the *right-click-select* inputs, syntactic correctness and consistency is vital. This step is grounded on the described syntactic configuration patterns. For the purpose of our proof-of-concept prototype we have hard-coded these relationships, which means that currently, the modeller maintains syntactic consistency manually by assigning related joins to splits or events to configurable functions. But it is also possible to assist this process with intelligent algorithms, e.g., for finding related joins of splits.
4. *Ensure Pragmatic Quality* – The "look" of the CEPC after performing a configuration task can be manipulated by *templates* within ARIS or physically shifting objects to make the model easily understandable. In this case, pragmatic quality merely means how visually appealing the modeller wants to represent the CEPC. This is not controlled within the project.

Finally, *Commit Configuration* completes the task by saving the configured CEPC into an "executable" EPC. Once the conceptual models are configured to represent a mirror of the organisation's processes (by repeating the three steps though all the organisation's business processes), the next remaining requirement is to execute this set of configured EPCs. This step would make actual configuration changes to the ES corresponding to the set of configured EPCs.

SUMMARY AND OUTLOOK

In this paper we have argued that it is of paramount importance to configure ES by means of conceptual models that are strictly bound to the ES' application logic. This need arises from the fact that it is questionable that ES vendors will eventually be able to deliver generic ES using functionalist approaches. Rather, we argue, that they should put major efforts into strictly binding ES functionality to conceptual models and provide sufficient means for the efficient adaptation of the conceptual models describing or rather specifying their software.

It is thus our aim to contribute to the understanding of model configuration, and more specifically, of generic situations that can occur during model-driven process configuration. In order to develop an understanding of such situations, we have outlined semantic and syntactic configuration patterns and briefly discussed their realisation in a prototypic environment.

We envision certain extensions to the described approach. First, we wish to explore which additional configuration patterns arise from using sophisticated workflow languages instead of EPCs. These new configuration patterns are especially important within the more technical parts of a configuration project. We also like to explore additional means of assisting syntactical configuration such as algorithms. Finally, we have to examine closer the relationships between the patterns of the syntactic and semantic dimensions in order to better understand process model configuration.

REFERENCES

1. Aalst, W. M. P. v. d., Hofstede, Arthur H. M. ter, Kiepuszewski, B. and Barros, A. (2003) Workflow Patterns, *Distributed and Parallel Databases,* 14(3), 5–51.
2. Akkermans, H. and van Helden, K. (2002) Vicious and Virtuous Cycles in ERP Implementation: A Case Study of Interrelations between Critical Success Factors, *European Journal of Information Systems,* 11(1), 35–46.
3. Barnard, C. I. (1938) *The Functions of the Executive,* Harvard University Press, Cambridge, MA.
4. Becker, J., Delfmann, P., Knackstedt, R. and Kuropka, D. (2002) Konfigurative Referenzmodellierung, In *Wissensmanagement mit Referenzmodellen. Konzepte für die Anwendungssystem- und Organisationsgestaltung,* (Eds, Becker, Jörg and Knackstedt, Ralf), Heidelberg, Germany, pp. 25–144.
5. Boje, D. M., Gephart Jr., R. and Thatchenkery, T. J. (Eds.) (1996) *Postmodern Management and Organization Theory,* SAGE Publications, Thousand Oaks, CA.
6. Brehm, L., Heinzl, A. and Markus, M. L. (2001) Tailoring ERP Systems: A Spectrum of Choices and Their Implications, In Proceedings of the *34th Hawaii International Conference on System Sciences (HICSS),* Hawaii, IEEE.
7. Burrell, G. and Morgan, G. (1979) Sociological Paradigms and Organisational Analysis: Elements of the Sociology of Corporate Life, Heinemann, London.
8. Chia, R. K. G. (1996) *Organizational Analysis as Deconstructive Practice,* Walter de Gruyter, Berlin et al.
9. Cyert, R. M. and March, J. G. (1963) *A Behavioral Theory of the Firm,* Prentice-Hall, Englewood Cliffs, NJ.
10. Dalal, N. P., Kamath, M., Kolarik, W. J. and Sivaraman, E. (2004) Toward an Integrated Framework for Modeling Enterprise Processes, *Communications of the ACM,* 47(3), 83–87.
11. Gephart Jr., R. (1996) Management, Social Issues, and the Postmodern Era, In *Postmodern Management and Organization Theory,* (Eds, Boje, David M., Gephart Jr., Robert and Thatchenkery, Tojo Joseph) SAGE Publications, Thousand Oaks, CA et al., pp. 21–44.
12. Gibson, C. F., Singer, C. J., Schnidman, A. A. and Davenport, T. H. (1984) Strategies for Making an Information System Fit Your Organization, In *Management Review,* Vol. 73, American Management Association, pp. 8–14.
13. Hatch, M. J. (1997) Organization Theory. Modern Symbolic and Postmodern Perspectives, Oxford University Press.
14. Holland, C. P. and Light, B. (1999) A Critical Success Factors Model for ERP Implementation, *IEEE Software,* 16(3), 30–36.
15. Klaus, H., Rosemann, M. and Gable, G. G. (2000) What Is ERP? *Information Systems Frontiers,* 2(2), 141–162.

16. Koontz, H. and Cyril, O. D. (1968) *Principles of Management: An Analysis of Managerial Functions*, McGraw-Hill Book Company, edition 4, New York et al.
17. Lindland, O. I., Sindre, G. and Sølvberg, A. (1994) Understanding Quality in Conceptual Modelling, *IEEE Software*, 11(2), 42–49.
18. Lucas Jr., H. C., Stern, L. N., Walton, E. J. and Ginzberg, M. J. (1988) Implementing Packaged Software, *MIS Quarterly*, 12(4), 536–549.
19. March, J. G. and Simon, H. A. (1958) *Organizations*, John Wiley & Sons, New York et al.
20. Markus, M. L., Petrie, D. and Axline, S. (2000) Bucking the Trends: What the Future May Hold for ERP Packages, *Information Systems Frontiers*, 2(2), 181–193.
21. Merriam-Webster (2003) Merriam-Webster's Collegiate Dictionary (Online Version Available under http://www.m-w.com), Merriam-Webster, edition 11, Springfield, MA.
22. Rosemann, M. and Aalst, W. M. P. v. d. (2003) A Configurable Reference Modelling Language, Centre for IT Innovation, Queensland University of Technology, Brisbane, Australia, to appear in *Information Systems*.
23. Scheer, A.-W. (2000) *Aris - Business Process Modeling*, Springer, edition 3, Berlin et al.
24. Soffer, P., Golany, B. and Dori, D. (2003) ERP Modeling: A Comprehensive Approach, *Information Systems*, 28, 673–690.

[*] The research results presented in this paper have been funded by SAP Research and Queensland University of Technology with the project “Modelling Configurable Business Processes”. SAP is a trademark of SAP AG, Walldorf, Germany.